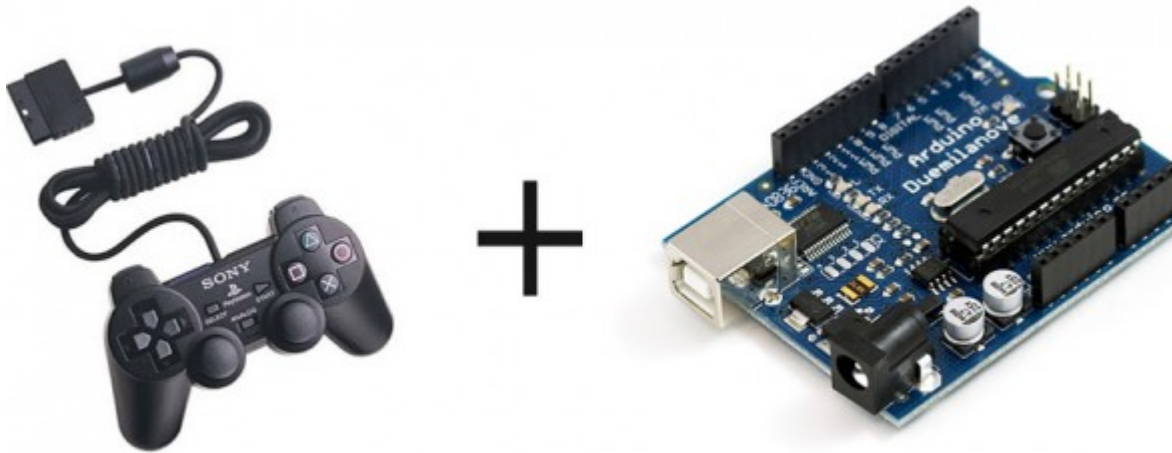


Arduino Playstation 2 Controller Library Troubleshooting Guide

Posted in [Arduino Libraries](#) by [Bill](#)

27 Mar 2011



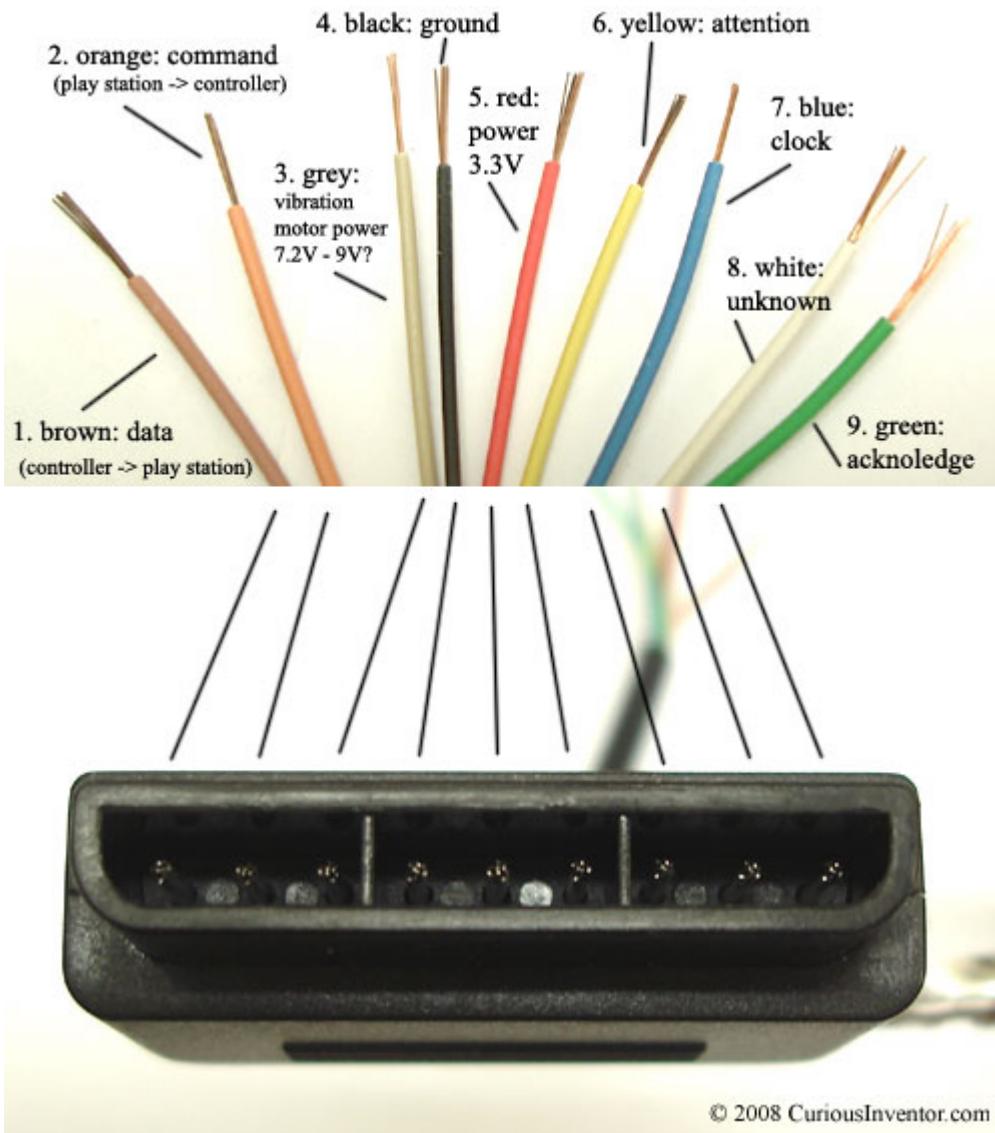
Ok, if you are here, you are having trouble getting your Arduino talking to your PlayStation 2 Controller via my [Handy Dandy PS2X Library](#). Well, never fear, I stand by my leet coding abilities (Ha! who am I kidding? Really though, the library works, there's tons of examples using it.) and I'm here to help. The problem is there's not much I can really do unless you live near North Florida and want to come pay me a visit. It's hard giving help over the internet! But maybe I can guide you in the right direction with this troubleshooting guide.

I'll walk you through some debugging steps, and you should follow through with me with DEBUG off until I tell you to enable it and using the example sketch unmodified. In-between attempts, power off the Arduino and controller for 10 seconds before testing it again. I'll bold the quick steps if you want to skim through.

A little background first. I have not personally reverse engineered the communications protocol a PS2 controller uses. That has been done by a few other cool guys [like this one](#). I used what they reported in writing the PS2X library. As far as I know Sony has never official published the communications protocol to the public, so reverse engineering efforts is all we have to go on. This has lead to a few problems; for example some controllers just don't behave like others. From the various sources around the web, the best chance of success comes from using 'Official' Sony controllers, and not the cheap knock-offs, 3rd party controllers. There's also some links with wireless controllers not working well either. Personally I have tried a wired Sony controller, and a few wireless Guitar Hero Controllers all with success, powered at 5V and no resistors.

Therefore the first thing to consider is what controller are you using? Can you try another brand, model?

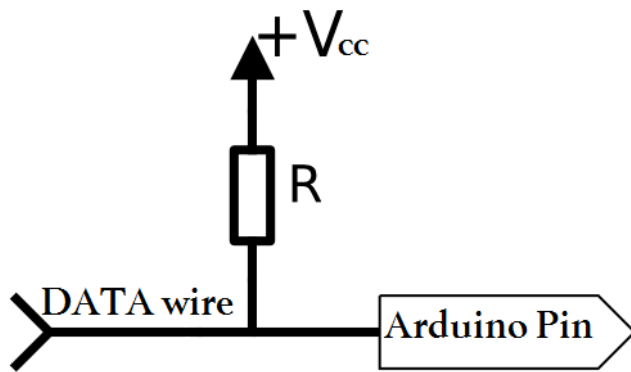
Ok, now that we are past history of controllers, I'll tell you this: 90% of the problems people have end up being related to wiring. I can't look at your setup, so you are going to have to use your best judgment. **Does the wiring look sound?** Controller wires are thin and small and tend to break easily. During prototyping, I cut the ends off my wired controllers, stripped 1/4" insulation off and **heavily** tinned the wires to create solid 'pins' that could be pushed into the female pinheader on the Arduino. This method worked, but was fragile. Later on I bought a few \$5 PS2 controller extension cables and cut off the female receptacle end. I soldered the wires from the plug to PCB board and that worked well also.



Don't trust the color scheme on the wiring picture. It can vary! Instead, use a voltmeter and test continuity between pin positions in the connector and the bare wires. I've had a few people report odd color schemes, and a few not realize what end of the connector they were looking at and got the wiring completely backwards. **A quick check is to power up the controller, and pushing the 'Analog' button. Either the LED should have been on when it powered up, or it should light up when you press the button.** If not, the controller isn't getting power and you probably have the wiring reversed.

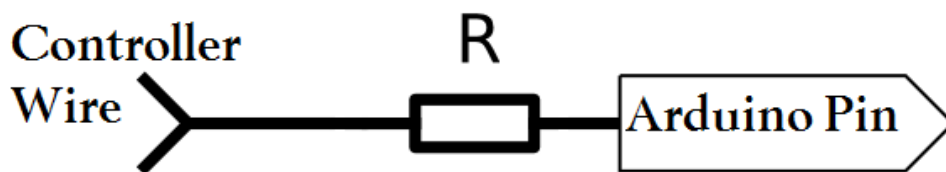
Alright, so you trust the wiring job, what next?

There are discussions over whether or not the controllers are tolerant of 5V logic. I've never had one burn out, so I believe they can handle "receiving" 5V, but they still won't 'send' 5V to the Arduino. It seems to me the one fix that seems to work the most is **adding a 10k resistor between DATA (brown wire, or pin 1 on the controller port) and VCC (try both 3.3V and 5V in that order) on the Arduino.**



This will help ‘pull-up’ the data line. Also there’s disagreement on what voltage to use to power the controller. A PlayStation powers the controller at 3.4V. I have always powered my controllers at 5V and had no problems, but it is risky. Controllers are so cheap on ebay though, why not try it? **Try different voltages for power, switch between 3.3V and 5V, and for each voltage try it with and without the pull-up resistor on the DATA line.**

One person claimed his controller wouldn’t work without level shifting all his wires, which means he just **added resistors inbetween the controller’s pins and the Arduino.**



The numbers he reported success with is as follow:

1. 5V = 180 ohm, voltage drop of 1.85V (3.15v to the controller)
2. ATT = 5.6K ohm, voltage drop of 1.08V (3.92 to the controller)
3. CLK = 5.6K ohm, voltage drop of 1.15V (3.85 to the controller)
4. CMD = 860K ohm, voltage drop of 1.39V (3.61 to the controller)

so you can give that a try as well. Though really, ignore the resistor on 5V and just connect the red power wire to 3.3V.

Some people report issues using pin 13 on the Arduino for CLK. **Try moving the CLK line to another pin, and editing the example sketch to match.**

If you have gotten this far and still no luck, you might just be out of luck. Controllers can go bad and some people just aren’t any good at wiring. But, there’s one more thing we can try messing with.

There’s a value in the ‘PS2X_lib.h’ file that governs the speed of the bus to the controller. It’s called ‘CTRL_CLK’ and you can find it by looking for this line:

```
#define CTRL_CLK    4
```

The PlayStation 2 talks to its controllers at 500kHz, or a value of ‘2’ in my library. Arduino tends to have issues setting a value that low, so by default I have it set at 4. **You can try using 2 instead, and I’d also try using some higher number for a slower bus speed. Go from 2-20 and even 200 to see if you can get the controller talking.** Remember to save the .h file every time you edit it, and re-compile the sketch.

Still no luck? Man you are killing me. I've dropped all my suggestions into this guide, so there's not much else I can say even if you ask really nicely. Florida is nice year round, so feel free to come visit me and I will help you in person. Otherwise, you can try to enable debug. It won't really help diagnosing wiring problems (or bad controllers) because all you will see is all FF's or 0's. But if you are getting some values, they may be something else going on. Drop the output in the [Support Forum](#) and I'll see if I can help.

My last fleeting thoughts include removing the checks in the example. At the end of the setup function (after the switch statement and before the closing curly bracket '}'), add the following code:

```
error = 0;
```

```
type = 1;
```

This will bypass the error checking and run the code anyway. What do you get? Try pressing reset on the Arduino without disconnecting the controller. Anything?